A middleware protocol for time-critical wireless communication of large data samples

Jonas Peeck*, Mischa Möstl*, Tasuku Ishigooka†, Rolf Ernst*
*TU Braunschweig, Institute of Computer and Network Engineering, Braunschweig, Germany
{peeck|moestl|ernst}@ida.ing.tu-bs.de
†Hitachi Ltd., Research & Development Group, Ibaraki, Japan
tasuku.ishigoka.kc@hitachi.com

***Abstract*—We present a middleware-based protocol that reliably synchronizes large samples consisting of multiple frames efficiently and within application level QoS requirements over a lossy wireless channel. The protocol uses a custom retransmission scheme, exploiting the latency requirements on sample level for frame level scheduling. It can be integrated into the popular DDS middleware. We investigate some technical limits of such a protocol and compare it to existing error protocols in the software stack and in the wireless protocol and combinations thereof. The comparison is based on an Omnet++ simulation using an established wireless channel error model. For evaluation, we take a use case from automated valet parking where infrastructure data provided via a wireless link augments in-vehicle sensor data. The use case respects the related safety requirements. Results show that the application awareness of the presented protocol, significantly improves service availability by transmitting data efficiently in time even under higher frame error rates.**

***Index Terms*—automated driving, V2X, wireless, real-time, reliability, sample, sensor data, DDS**

I. Introduction

Sending data with real-time constraints via wireless channels, i.e. channels with non-negligible loss rates, has always been challenging. Extensive research has been done addressing such setups to provide real-time bounds for individual Media Access Control (MAC) layer frames e.g. [1]–[5]. However, these works suffer from the fact, that they are agnostic about real-time requirements of higher layer data structures, e.g. image data which is larger than an individual MAC layer frame's payload. We refer to such data as application data, or simply **samples**, as it is done by the popular Data Distribution Service (DDS) [6]. Samples must be fragmented into frames, in order to be transmitted, however, the real-time requirement of a sample can not be straightforwardly decomposed into frame requirements. This is due to the fact, that it is sufficient that all fragments of a sample reach the destination before the deadline of the sample. Hence, flexible strategies for *retransmission* of lost frames (or acknowledgements) are possible. Furthermore, management of scarce channel resources can be optimized, based on the sample-level requirements. Thus, it can be avoided to utilize the channel for frames which contain fragments of a sample that would violate the sample's timing requirement anyway. Such situations occur if more participants utilize a channel and frame losses are high. Thus, by limiting the load injection or even aborting a sample transmission, channel utilization can further be relaxed. This allows other participants to use the channel with shorter arbitration delays.

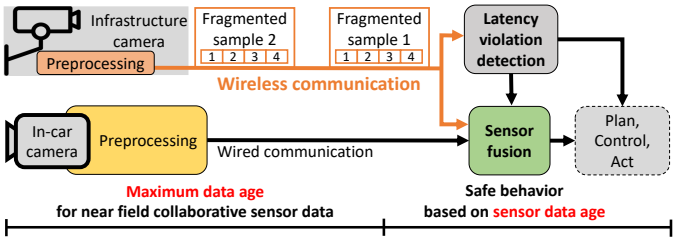


Fig. 1: Integration of external sensor data into the car's sense-to-act data processing chain.

For illustration, we provide a use-case, which we also use for evaluation later on. Fig. 1 sketches an application from automated valet parking [7], [8], where the video data from an infrastructure camera augments the in-vehicle sensors to obtain images from different directions helping to detect hidden objects. The video stream consists of a periodic sequence of camera frames. The camera frames are the data objects of concern, i.e. the samples. Samples are generated with a **sample period** (P_S), in our case the camera frame rate. While there are certainly approaches to sensor fusion with uncertain and incomplete camera data, we want to separate concerns for predictable sensor fusion results. Furthermore, samples and sample sequences are usually compressed in order to achieve sufficient image resolution, such that the loss of one sample also leads to the non-usability of others due to data dependencies between compressed frames. Therefore, each sample is only valid if all its camera frame data (i.e. pixels or compressed data) are available and a video sequence is only valid if all its camera frames are available and valid.

As the sensor fusion must not work on outdated data, the permitted maximum sensor data age is limited by the sensor fusion, w.r.t the car's driving context. The related age requirement is defined as **sample deadline** (D_S) that determines the permitted sample latency. In our use case, we define $D_S = P_S$, whereby P_S is $100\,\mathrm{ms}$ [9]. The **sample size** (S_S) is determined by camera resolution and fusion requirement and typically ranges between tens of kB to several MB. Each sample is fragmented into many frames for transmission at the MAC layer of a wireless network, e.g. IEEE 802.11 media frames.

As stated above, there is no need to translate sample deadlines to MAC layer frame periods and frame deadlines. However, sample deadlines provide slack for access timing and error correction of the MAC layer frames of a sample. On the other hand, sample deadline violation is a safety case, because the augmented sensor system supports early object detection, which can be used to allow faster vehicle motion. Its violation is a reliable indicator for unacceptable data quality to immediately fall back to in-vehicle sensors and reduce vehicle speed. The fall-back concept emphasizing data integrity makes the augmented automated service fail-operational [10] w.r.t. the wireless communication.

The presented use-case can also be seen as a representative of the class of collaborative sensing, which is intended to flatten the risk curve for automated SAE level 3+ driving [11]. Collaborative sensing is part of current research roadmaps [12], [13] and combines high communication requirements [13] with safety constraints. Since vehicle communication is an intrinsically lossy technology, the challenge is to keep fault tolerant high-bandwidth real-time communication up as long as possible, but reliably detect error situations to fall back to a safety layer. However, timing and reliability requirements of large data objects are not covered by state-of-the-art Vehicle-to-everything (V2X) software stacks, as they are only designed to support the upcoming standards, e.g. [14].

We present an application-aware protocol for samples that integrates into the DDS software stack. DDS is applied in the robot operating system framework (ROS) 2 where it is a core component and AUTOSAR Adaptive Platform [15]. DDS is a sample based publish-subscribe middleware which abstracts communication of application data by synchronizing samples between databases. The sample synchronization is subject to Quality of Service (QoS) parameters such as latency. However, DDS can neither give latency guarantees directly, nor can it efficiently control lossy wireless communication such that frame retransmissions respect a sample deadline [6].

Our Contribution is a protocol for latency-critical wireless synchronization of larger-than-a-frame data samples over lossy wireless channels and its evaluation. We used DDS communicating over 802.11p (a V2X standard) as an important practical use case. We demonstrate that awareness of application properties (object structure, rates, application deadlines) allows major improvements in wireless vehicular communication for time and safety critical applications. Rather than trying to optimize and adapt to the channel at runtime, our work provides a protocol with static parameters and a related safety concept to meet application safety requirements.

The paper is structured as follows: We start with a review of related work in wireless communication stacks as well as higher layer middlewares and protocols in section II. In section III, we take a deeper look into the wireless communication stack and its many protocol parameters and their modelling, as proposed in the literature. Next is the description of our sample-based protocol in section IV and its integration in the DDS middleware in section V. A large part of the paper is dedicated to an evaluation of the protocol in section VI. It is compared against the technical limits of the wireless channel protocol and against existing solutions. While the valet parking use case is taken as a basis for evaluation, the results are general for wireless sample communication. Conclusions are drawn in section VII.

II. Related Work

We first introduce relevant work w.r.t. V2X communication stacks before we review related work on sample-based middlewares and protocols.

A. Communication Stacks

V2X communication has been developed and standardised over many years. The main objectives are to improve road safety, efficiency and also to enable autonomous driving. There are two main communication standard groups for so-called Dedicated Short Range Communication (DSRC), which are summarized in the Wireless Access for Vehicular Environments (WAVE) protocol stack in the US [16] and the ITS-G5 from the European Telecommunications Standards Institute (ETSI) [17]. WAVE and ITS-G5 are very similar as the European ITS-G5 is derived from WAVE. Both protocol stacks are based on IEEE 802.11p as the physical and media access layer [18], which is a derivate of the WLAN standard. In contrast to standard WLAN, ITS-G5 and WAVE define their own protocols on the network layer, especially routing algorithms such as GeoNetworking in ITS-G5 or WAVE Short Message Protocol (WSMP) in WAVE, which do not employ IP as a protocol. The stacks contain a predefined set of messages representing either status information of cars or recognized events. Different message types are intended for predefined application groups. The currently most relevant message types are Cooperative Awareness Message (CAM) in ITS-G5 [19] and Basic Safety Message (BSM) in WAVE [20]. They are broadcasted periodically and contain status information like position, speed and movement direction of the transmitting vehicle to other traffic participants. As this information can help to drive proactively or even enable cooperative behavior, it is bounded to predefined data elements within the messages. Transmission of more complex data, such as large collaborative sensor data, as required for future autonomous driving applications are not applicable as the message format is fixed.

However, at least the WAVE stack allows to use Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) with IP as the network protocol without the need to use the predefined messages. While UDP alone does not recover lost datagrams (e.g. parts of a sample), TCP does retransmissions, however, it is not real-time capable in its standard version. TCP's retransmission scheme focuses on congestion control in the internet by introducing large waiting times after losses, based on the limitation of the maximum send and unacknowledged messages. In addition, TCP misinterprets bit error based losses as congestion issues, which worsens the problem.

More recently also cellular networks have been standardized for V2X communication. Instead of specifying a completely new network stack, C-V2X replaces the IEEE 802.11p access

technologies. Thus, the problems of the upper layers remain unresolved for the transmission of large samples.

We can conclude, that current state-of-the-art V2X wireless communication stacks do not only lack the capability of intelligent retransmission handling of large sample data, its transmission itself is not even addressed e.g. by a certain application group or a specified message type. Thus, we tackle this problem in this paper with the use of a dedicated DDS middleware, which is able to extend the DSRC WAVE protocol stack where necessary, by setting up on its UDP/IP stub.

B. Middleware and Protocols

As explained, the existing V2X stack infrastructure is unsufficient to manage reliability of real-time constrained sample data over a wireless channel. In the following we will review related work for applicability towards this missing protocol properties on different stack layers.

Because standard TCP is not suitable for wireless real-time applications, a number of different TCP variants, e.g. [21], [22], explicitly address wireless communication, and therefore, aim to provide high throughput under random losses. However, their efficiency is limited to a frame loss rate of a few percent. Other work tries to build protocols directly on UDP [23], [24], but only requirements of individual frames and not of entire samples are considered.

With the introduction of ROS 2, the focus has shifted to DDS as a middleware specification [6]. However, peculiarities of wireless and hence lossy communication are not explicitly addressed. Therefore, various works present custom implementations on that specification. The DDS implementation in [25] for instance is intended for embedded and resource constrained environments yet does only consider wireless but non real-time communication. A real-time setup has been studied in [26], but only for reliable wired communication.

A DDS implementation in the field of Wireless Sensor Networks (WSNs) is presented in [27], but does not consider the tight data age and latency requirements of automotive sensor fusion. Moreover, the category of WSNs describe the application domain, rather than specific techniques on realiable or real-time data transfer. Thus, each WSN which aims to transmit large sensor data is bounded to the state-of-the-art stack limitations reviewed in this paper, which origin in the focus on single frame transmissions. In general, todays research on WSNs is focused on different domains with different requirements. Real-time setups for WSNs used in industrial applications are build up on a non-WLAN but TDMA-based approach in order to guarantee latencies of single frames instead of samples [3], [4]. Moreover, these setups are static or at most include a low degree of dynamics and are thus not applicable for the highly dynamic automotive environment with high frame loss rates. An analysis for a TDMA based scheduling is proposed within the Sensor Network Calculus [28], but do not consider the impact of frame losses. Another protocol for general mixed-criticallity CPS applications is AirTight [5]. The fundamental difference to our sample-based problem statement is, that AirTight focuses on scheduling independent packets types, whereas the packet deadlines are smaller or equal to their period. In contrast, our protocol addresses scheduling of multiple frames corresponding to the same packet type, i.e. a sample, with the goal to transmit each frame at least once until a shared deadline.

As there are no suitable protocols for reliable real-time sample transmission at the transport and middleware levels, an alternative candidate might be the MAC layer. Especially frame aggregation with retransmission in WLAN networks has proven to increase throughput by reducing average arbitration times [1], [2], [29], [30]. However, frame loss is inherently included within the MAC layer, which only permits an upper bound on retransmission attempts that is likely to be exceeded by burst errors and unfavorable channel conditions. This applies to WLAN and cellular communication [31], [32]. Thus, a reliable communication must always include higher-level retransmission mechanisms to cover losses. As we will show in our evaluation, the combination of sample unaware MAC and higher level reliability mechanisms leads to unvafourable cross-layer effects violating sample deadlines as well as leading to unpredictable interference of other sample transmissions.

In summary, a reliable and real-time constrained transmission of samples over a lossy wireless channel in an automotive setup is still an open problem. Therefore, we provide (to the best of our knowledge) the first protocol that exploits the application requirements for improved inter-sample frame scheduling so that sample latency constraints can be guaranteed even at high error rates.

III. Stack model

As motivated in the previous sections, the current V2X software stacks are inappropriate to adequately support the sample-based use case from section I, as appropriate mechanisms to protect time-sensitive samples are not available. Before developing our middleware protocol, addressing the challenges shown in the previous section, we will introduce the properties of a wireless channel for mobile vehicular applications in the context of sample-based reliable real-time applications. On top of the lossy wireless channel, we concentrate on the IEEE 802.11p MAC-protocol as the underlying layer of a middleware protocol. We describe our model abstractions for the channel and channel access, which we use in our simulation-based evaluation in section VI.

A. UDP/IP for high-level protocol development

UDP is a connectionless protocol for sending data with minimal overhead to another application. It follows a real-time compatible approach, as it does not consist of timing-unpredictable components, as TCP does. In detail, it passes data directly to lower layers. Hence, in combination with its small protocol overhead of $8\,\mathrm{B}$ it builds the basis for various higher-level sample-based middleware protocols like DDS, as it is the only widely supported alternative to TCP for standard IP communication. In addition to the UDP overhead the IP

layer adds $20\,\mathrm{B}$ of protocol overhead, which is needed for routing.

B. The Media Access Control protocol

On the lower layers of a communication stack, physical transferral of data between devices, including shared medium access, has to be organized. A typical approach in wireless communication is unsynchronized arbitration. This is also the case in ITS-G5 and WAVE, where IEEE 802.11p [18] defines a Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) MAC protocol.

The basic arbitration principle is that before sending a frame, the channel must be free and each sender must wait for a random time before beginning to send. The random waiting time has the goal of preventing collisions of senders that want to start transmitting at the same time. A channel is considered free for a transmission after waiting for a certain number of slots N_{slot} with the slot time t_{slot}. The value of N_{slot} is drawn uniformly random from a Contention Window (CW). We refer to the timespan from the start of waiting N_{slot} slots until the start of message transmission as the arbitration time t_a.

A frame transmission fails, if either fading effects corrupt the signal or frames from different senders collide. Successful reception is immediately answered by the receiver with a privileged acknowledgment (ACK). In case the transmission fails, the MAC protocol offers the possibility to retransmit an unacknowledged frame starting with a newly generated N_{slot} value drawn from CW. To avoid an overloaded channel, the Binary Exponential Backoff (BEB) method is applied. In BEB the CW value is doubled every time a frame is not ACK'ed on the MAC layer, even if the frame loss is due to bit errors instead of a collision. This approach can lead to head-of-line blocking, within the same queue up to appr. $30\,\mathrm{ms}$ [32], which is clearly not compatible to D_S requirements of e.g. $100\,\mathrm{ms}$ in combination with a large frame count of a sample. Thus, we configure our setup to skip the BEB, which is permitted by [18]. Note, that in an overloaded channel to which the BEB is tailored, service augmentation as considered by our use case is not available anyway, since a late sample is just as useless as a faulty one. The car would have to fall back to the safe low-speed driving service with on-board sensors. Here it is important to underline, that a time-sensitive wireless service always **depends on a feasible channel load**.

Considering the remaining maximum of 7 retransmissions for a fixed CW, there are two main implications within our resource-constrained and time-sensitive sample transmission setup. First, a UDP-based protocol, where datagrams are acknowledged with an ACK-in-UDP datagram, would experience a wide Round-Trip Time (RTT) interval, however, in a time-sensitive setup a low or at least deterministic RTT is favourable. In detail, if the possible RTT increases, speculative retransmissions of datagrams might become necessary, as the sender is under time pressure w.r.t. D_S. This can lead to situations, where data is retransmitted although it is not lost, e.g. if an ACK datagram is lost after a successful transmission or the frame is still pending within the sender's MAC queue. Second, a middleware protocol loses the ability to control the injected channel load, as one transmission on the UDP level might lead up to 8 transmissions on the MAC layer. Thus, at bad channel conditions a sample transmission can lead to an overload situation and the sample transmission might still fail. As a consequence, resources for other traffic participants are blocked. Consequently, we disable retransmissions on the MAC layer, which is also permitted by the standard. As a beneficial side-effect it omits MAC layer ACKs, which cause about $40\,\mu\mathrm{s}$ overhead (depending on the configured data rate) per successful frame transmission and can also get lost.

Fig. 2 illustrates the resulting timing of channel arbitration and frame transmission. To simulate the individual waiting time of a single frame, we assume an average arbitration time $\bar{t}_a$ as a channel and environment parameter. It is the average time a frame has to wait for transmission after the slot number N_{slot} is drawn from CW. We assume this parameter to be constant, assuming a constant channel load over a longer period of time, especially a sample period. Please, note that $\bar{t}_a$ depends directly on the number of competing channel accesses, so it directly expresses the load that other senders are injecting to the channel. As the number of slots to wait is drawn from a uniform distribution, we draw the arbitration time t_a for a particular frame from a uniform distribution around $\bar{t}_a$, which we define as follows:

$$t_a = rand(0, 2) * \bar{t}_a \tag{1}$$

In this arbitration time, additional timing effects e.g. due to the underlying OFDM signalling are already abstracted.

When transmitting multiple messages, the average time between two transmission attempts is then the average blocking time $\bar{t}_b = \bar{t}_a + d_{\text{fr}}^{\text{msg}}$, i.e. the average arbitration time plus the actual transmission duration of the frame on the channel after successful arbitration (cf. Fig. 2). The transmission duration $d_{\text{fr}}^{\text{msg}}$ depends on the data rate of the wireless technology R_b as well as the bit size of the message frame S_{fr}^b: $d_{\text{fr}}^{\text{msg}} = \frac{S_{\text{fr}}^b}{R_b}$.

Besides WLAN also cellular technologies are developed for V2X communication. Even if this is not the scope of this paper, they could be modeled similarly. As their arbitration method is based on fixed timeslots within a global schedule, we have an even more deterministic arbitration time, to which the distribution of $\bar{t}_a$ or $\bar{t}_b$ would have to be adapted.

C. Properties of the lossy wireless channel model

The faulty transmission of a frame can be categorized into two classes: Arbitration conflicts, i.e. collisions represented by a frame error rate (FER), and effects due to properties of the wireless channel in general, which are typically represented in a vehicular environment by a Bit Error Rate (BER) [33], [34]. With respect to the CSMA/CA arbitration principles of IEEE 802.11p, the amount of collisions is related to the utilization of the channel, i.e. its multi-user access rate [35]. A high number of collisions, implies an overloaded channel which is not capable to transfer large collaborative sensor data anyway. Hence, we do not consider these cases in our error model, as well as we do not consider sudden shadowing

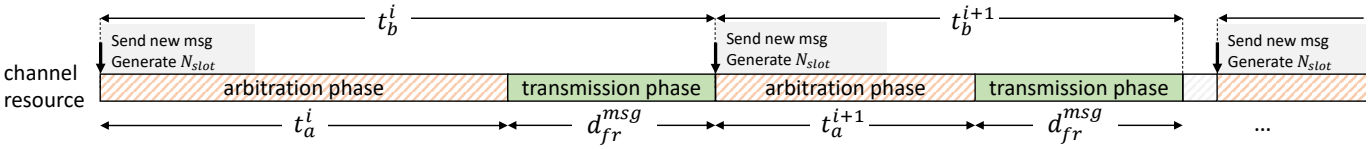


Fig. 2: High level view on the MAC arbitration scheme used for our arbitration model.

effects, which may even lead to a short term disconnection and thus to burst error lengths larger than D_S. As explained above, such cases must be detected and will lead to termination of collaborative sensing. Rather, we focus on evaluating our protocol in a variety of individual static error setups, where we can determine the maximum tolerable BER. Thus, our work explicitly addresses feasible channel conditions, in which the second error class of bit error related frame error dominates. Root causes for bit-level errors are fading effects like reflections, Doppler shift, etc. that influence the signal-to-noise ratio. Physical layer forward error correction (FEC) techniques are used to recover single bit errors by adding redundancy, however, at certain channel conditions full correction is not possible. Thus, we refer to the BER as the residual BER, i.e. the BER which remains after passing the physical layer's FEC. In consequence, the FER is determined based on the BER E_b and the bit size of the message frame S_{fr}^b [33]:

$$E_{\text{fr}}^{E_b}(S_{\text{fr}}^b) = 1 - (1 - E_b)^{S_{\text{fr}}^b} \tag{2}$$

We assume that a frame is faulty when at least one bit is corrupted. This channel model is an approximation because the physical level FEC is more effective at larger frame sizes. Since this size dependency influences all higher level protocols in a comparable way, we omitted this effect for simplicity.

The combination of $\bar{t}_a$ and BER limits the amount of data which can be transmitted timely, as larger frame sizes get lost at higher rate and smaller frame sizes lead to more arbitration attempts for the same amount of payload data.

IV. Middleware Protocol

In this section, we introduce our Wireless Reliable Real-Time Protocol (W2RP), addressing the requirements of our automotive use case from section I. Before going into protocol details in section IV-B and section IV-C, we review technical limits of such a protocol for our channel model in section IV-A.

A. Technical limits of a realiable real-time sample transmission protocol

The main goal of a protocol in the context of our work is to comply to the application's QoS requirements, which are the sample deadline D_S and the sample size S_S. This means, that the additional input from infrastructure sensors is only useful for the car's processing chain if it arrives timely w.r.t. D_S and e.g. for camera data with a sufficient resolution i.e. S_S. However, compliance to these requirements face certain channel limitations, which we have modelled by $\bar{t}_a$ and $E_{\text{fr}}^{E_b}$. W2RP is mediating between the requirements and the channel resources, w.r.t. an intelligent transmission schedule. To be able to evaluate a protocol properly, we first define the technical limits in relation to the requirements and the channel model. In doing so, we address the question of which requirements are maximally manageable by a *generic protocol* under a given channel model. This enables us to evaluate W2RP according to how close it gets to these bounds.

First, a sample with the size S_S is split into a certain number of fragments N_S^f depending on a fragment size S_f:

$$N_S^f = \lceil \frac{S_S}{S_f} \rceil \tag{3}$$

As we define S_f as a constant value within a sample and it is the relevant factor for the BER-based frame error rate $E_{\text{fr}}^{E_b}$ (cf. eq. (2)), we do not consider sending multiple or a varying number of fragments within one message. However, the last fragment of a sample might be reduced in size. Following the average blocking time at the MAC layer, the first limit is given to the minimum feasible deadline requirement:

$$D_S^{\min} \geq N_S^f * \bar{t}_b \tag{4}$$

Here, each fragment has to be transmitted at least once within D_S, which depends on the average blocking time as well as the number of fragments the sample is divided into.

Decreasing N_S^f by increasing S_f would relax this condition, however, this is in conflict with the bit error rate model. It increases the probability to lose the message, thus adding the need for an additional retransmission. The maximum number of retransmissions is the difference between the maximum number of messages ready to send within D_S and the number of different fragments per sample. It has to be at least greater or equal to zero (w.r.t. the loss-free case):

$$N_S^{\text{retr,max}} = \lfloor \frac{D_S}{\bar{t}_b} \rfloor - \lceil \frac{S_S}{S_f} \rceil \geq 0 \tag{5}$$

From eq. (5) we can now derive the maximum tolerable message error rate $R_{\text{msg}}^{\text{max,err}}$, which is the relation between the maximum number of retransmissions and the maximum number of messages ready to send within D_S:

$$R_{\text{msg}}^{\text{max,err}} = \frac{N_S^{\text{retr,max}}}{\lfloor \frac{D_S}{\bar{t}_b} \rfloor} = 1 - \frac{\lceil \frac{S_S}{S_f} \rceil}{\lfloor \frac{D_S}{\bar{t}_b} \rfloor} \geq E_{\text{fr}}^{E_b} \tag{6}$$

It can be also represented as one minus the minimum ratio of necessary successful message transmissions. Following this, $R_{\text{msg}}^{\text{max,err}}$ has to be greater than or equal to the frame error rate in order to be able to retransmit all lost fragments within D_S.

These technical limits define the outer bounds of what any fragmentation-based protocol is able to handle within

the given model assumptions. In reality a protocol is not optimal. Especially within real-time requirements a protocol cannot wait for indefinitely for reception of ACK information. Thus, it has to act speculatively up to a certain degree. As a consequence, a fragment can be sent unnecessarily twice in a backward error correction (BEC) approach, e.g. if the corresponding ACK was dropped by the channel or is strongly delayed. Due to this loss of information and round trip delays within a real-time constrained BEC approach, uncertainty is inherently present, and has to be minimized. In order to address this issue, we start with its formalization. We call the number of unnecessarily sent fragments $N_S^{\text{f,unnec.}}$ in relation to the overall number of sent fragments $N_S^{\text{f,total}}$ the efficiency Eff_S of a protocol. It is defined per sample and represents a major performance parameter for the protocol:

$$\text{Eff}_S = 1 - \frac{N_S^{\text{f,unnec.}}}{N_S^{\text{f,total}}} \leq 1 \tag{7}$$

In detail, an unnecessarily sent fragment is one which is retransmitted and a previous transmission already succeeded. As this blocks resources for necessary retransmissions the efficiency directly affects the protocol's ability to match the technical limits. For retransmission decisions, efficiency is most relevant w.r.t. the maximum tolerable message error rate $R_{\text{msg}}^{\text{max,err}}$. Thus, the effective maximum tolerable message error rate $R_{\text{msg,eff}}^{\text{max,err}}$ is reduced by the factor of the efficiency:

$$E_{\text{fr}}^{E_b} \leq R_{\text{msg,eff}}^{\text{max,err}} = R_{\text{msg}}^{\text{max,err}} * \text{Eff}_S \tag{8}$$

Consequently, with an increasing efficiency, a protocol gets closer to the technical limits. Note, that besides the violation of the technical limits, some sample transmissions can still succeed due to the statistical behavior of the wireless channel and the distribution of the arbitration time, however, a reliable transmission is not possible anymore.

B. Utilizing DDS concepts

As we address the sample-based DDS middleware, which allows specifying QoS constraints, as the target platform to integrate our protocol algorithm, we first present existing structures and concepts we can build up on. A closer look reveals, that the underlying Real Time Publish Subscribe Protocol (RTPS) [36] – the wire-protocol of DDS – is responsible for the concrete real-time behavior and implements relevant protocol structures and mechanisms. First, we comply with the RTPS terminology of *fragments* describing parts of a sample to be sent within lower layer frames. RTPS uses bitmap-based ACKs for BEC, i.e. one bit of the bitmap signals the reception (1) of a fragment or its loss (0), starting from a reference fragment sequence number (FSN). As DDS also uses this concept to reduce the amount of ACKs to send, we use its low-cost redundancy to protect against lost ACK messages. Bitmap-based redundancy implies the risk to negatively acknowledge a fragment within two consecutive negative ACKs (NACKs) based on the same state of information. The consequence can be an unnecessary retransmission. RTPS protects itself with the so-called *NACK Supression Duration*, in which a fragment cannot be negatively acknowledged multiple times. We refer to it as the *NACK Guard* in the following. Besides the bitmap concept, RTPS uses heartbeats to trigger ACKs. Those heartbeats contain the highest yet sent FSN of a sample, which we use to signal a receiver the upper bitmap bound to provide. We append this information to each fragment message.

C. Mechanisms of W2RP

Based on the concept of fragmentation and bitmaps, we now provide strategies for the concrete timing and scheduling of fragment retransmissions that neither DDS nor RTPS offer. The goal is to meet the D_S requirements of a sample as well as to limit the load injection into the channel, which is necessary for cooperative resource management on a higher organizational level. Therefore, the focus is on *when* and *which* fragments should be (re-)transmitted. Fig. 3 shows a basic example of a W2RP writer and reader that synchronize their databases, i.e. their sample storages. Note, that the database on the writer side stores only up to one sample, thus its lifespan is equal to P_S.

1) Smoothing round trip times and limitation of load injection: As we have motivated in section III-B an efficient W2RP layer loss detection benefits from a deterministic RTT, hence MAC layer retransmissions are disabled. A second influence leading to an unstable RTT are queuing effects at the MAC layer buffer. After the application passes a sample to the writer's database (1) (cf. Fig. 3), it is crucial, that the writer does not force a burst transmission, i.e. sending all fragments at once to the MAC layer buffer. Obviously, the first fragment in the buffer would have a much lower RTT than later ones. Hence, we shape the time between the transmission of two fragments with the fragment-shaping-time parameter t_f^{sh} (2). This way t_f^{sh} shapes lower $\bar{t}_b$ values at the middleware level and relaxes MAC queuing effects over the whole sample transmission especially for $t_f^{sh} \geq \bar{t}_b = \bar{t}_a + d_{\text{fr}}^{\text{msg}}$. Thus, the effective average blocking time of MAC access and W2RP $\bar{t}_{b,\text{eff}}$ is defined as follows:

$$\bar{t}_{b,\text{eff}} = max(\bar{t}_b, t_f^{sh}) \tag{9}$$

For $t_f^{sh} \leq \bar{t}_b$ we will see queuing effects at the MAC layer buffer, which drastically increase the RTTs. This can be considered as a (transient) channel overload in the context of our protocol parametrization w.r.t. maximum feasible $\bar{t}_a$ values. Note, that for $t_f^{sh} > \bar{t}_b$ we limit the protocol capabilities w.r.t. to a higher number of (re-)transmissions within D_S and thus to comply to a higher FER. However, this property guarantees the maximum load injection, i.e. resource allocation, of W2RP, which limits its impact to the $\bar{t}_a$ value of other sample communications and enhances a well-organized overall transmission.

On the receiver side all messages are directly answered which aims to keep the RTT low (3). To model processing on the receiver we add the static delay t_d. Note, that the t_f^{sh} shaping is indirectly forwarded to the receiver.

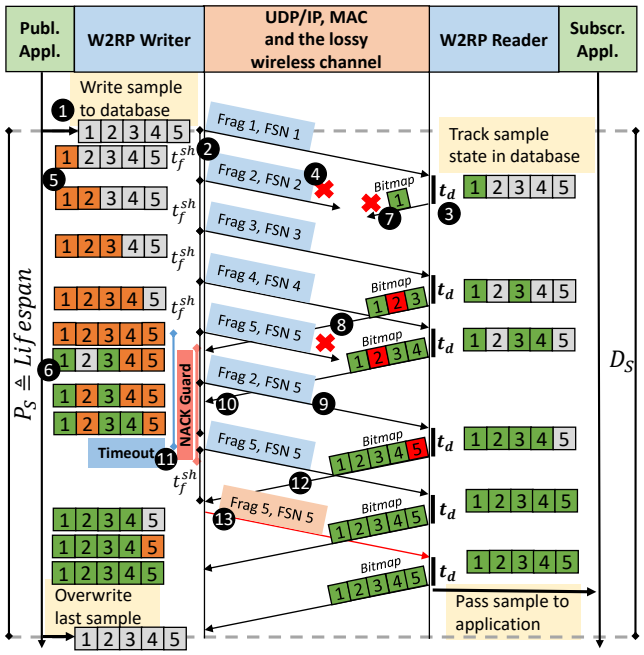


Fig. 3: Principles of W2RP for a sample transmission consisting of 5 fragments between a publishing and a subscribing application.

2) Fragment status tracking: Before describing which fragments to send, we will specify their transmission states stored in fragment status objects (FSOs) on both sides, to be able to derive scheduling decisions from them. A writer can apply the states *unsent* (grey), *sent* (orange) and the final state *acknowledged* (green) to its fragments. Each *sent* state is annotated with the corresponding *send timestamp*. The reader applies the states *unreceived* (grey) and the final state *received* (green). When sending a fragment message over a lossy channel, it can either get lost (4) or arrive successfully at the reader (3), where it is marked as *received*. The writer changes the states of its FSOs after sending a fragment to *sent* (5) or receiving a bitmap (6). The reception of a NACK turns the fragment into *unsent*, an ACK into the state *acknowledged*.

3) Loss detection mechanism and consequences for the scheduling: Now that we know *when* to transmit fragments, i.e. shaped with t_f^{sh}, we need to look at *which* fragments to transmit by the writer, based on their FSOs. This is essential, because only not yet sent fragments or lost ones should be sent. A retransmission of a successfully transmitted fragment degrades both the efficiency and thus increases latency. Therefore, the main idea is to **send fragments as fast as possible** w.r.t. t_f^{sh} and select the next fragment to send **based on the highest evidence** in the context of a necessary (re)transmission. Obviously, after a sample is written and its FSOs are initialized (1), we can start transmitting fragments safely. Therefore, we go through all available fragments in a circular loop based on their FSNs while receiving bitmaps from the reader in parallel. Lost bitmaps (7) are simply recovered by following ones (8), due to the low-cost redundancy mechanism. A fragment whose FSO is changed to *unsent* by a NACK is retransmitted within the circular approach (9). A first NACK of a fragment can be safely interpreted as a loss, the relevance of subsequent ones is evaluated by the NACK Guard (10), which we set to the RTT. Due to the circular approach the NACK Guard is mostly relevant when only few fragments are left unacknowledged. Otherwise, subsequent ACKs have more time to arrive.

NACKs are still the strongest indicator to trigger a retransmission, however, the last fragment to be sent can get lost, so that no bitmap will be created by subsequent transmissions. Therefore, we introduce a timeout (11), which we also set to the RTT. In case multiple timeouts are pending, we prefer the oldest one by its corresponding send timestamp. A timeout-based retransmission can only be triggered if no fragment is in state *unsent*. This way NACKs have priority over timeouts as they indicate a higher loss evidence. After the timeout the writer receives an additional NACK (12), due to the highest FSN sent within (9), forcing an unnecessary retransmission (13). Such an effect can only be seen at the end of a sample transmission (or at very high error rates) where timeouts trigger, because there are no more negatively acknowledged fragments to be sent. At this point the protocol speculates on a loss of the timeout message to keep the latency low. Note that with a larger S_S consisting of more fragments such a transmission is less relevant due to overall efficiency, but can have a positive effect on the sample's latency.

As the RTT is not explicitly known and can only be estimated, we introduce the *Selected Round Trip Time (SRTT)*, which is applied to the NACK Guard and the timeout. Lower values lead to more unnecessarily sent fragments, as the NACK Guard allows a status reset too early and timeouts might trigger before corresponding ACK arrivals. Higher values decreases the grade of speculation, however, especially fast timeout-based retransmissions at the end of a sample can decrease the sample latency. Since t_f^{sh} refers to the upper blocking time, we propose the following estimation for SRTT:

$$\text{SRTT} = 2 * t_f^{sh} + t_d \tag{10}$$

Thus, we address two blocking times (arbitration with transmission) and the receiver delay.

4) Early deadline violation detection and application feedback: In our setup, the deadline is reached when all pixels have been received. This can easily be detected by W2RP and used to trigger an "emergency operation" (fall-back layer) – as defined in Clause 3.40 and Fig.4 in ISO 26262-1:2016 — if the deadline is violated. In our case, the fall-back layer is to operate only on in-vehicle sensors at reduced vehicle speed. Such a safety concept is compatible to ISO 26262 and should be certifiable.

The protocol's sample transmission status can also be exploited to perform detection of *upcoming* deadline violations at both the receiver and sender side. This provides extra time for

fail-operational safety mechanisms, e.g. switching the sensor fusion to in-vehicle sensors only. In addition, early violation detection enables W2RP to stop sample transmission already before D_S in order to save channel resources for other communication participants. First, we apply t_f^{sh} to $\bar{t}_{b,\text{eff}}$, as the known minimum for the effective blocking time. Corresponding to the technical limit from eq. (4), we can analyze how many sending approaches $N_S^{f,\text{rem}}$ remain between now (t^{now}) and D_S, w.r.t. the start of sample transmission t_S^{start}. $N_S^{f,\text{rem}}$ should be larger than the number of unacknowledged fragments $N_S^{f,\text{unacked}}$, otherwise we can expect a deadline violation and notify the application in advance:

$$N_S^{f,\text{rem}} = \lceil \frac{D_S - (t^{\text{now}} - t_S^{\text{start}})}{t_f^{sh}} \rceil \geq N_S^{f,\text{unacked}} \tag{11}$$

This approach reliably detects an emerging timing violation, however in the worst-case directly at the deadline. In addition we observe the current effective error rate $R_{\text{msg,eff}}^{\text{cur,err}}$ under the assumption of stable channel model parameters within D_S. Through this estimate, which represents the channel state of the current transmission, we make a forecast about the success of the remaining fragments in relation to the remaining sample deadline budget. The current error rate is based on the difference between the maximum number of sent fragments $N_f^{\text{send,max}} = \frac{t^{\text{now}} - t_S^{\text{start}}}{t_f^{sh}}$ over the number of received non-duplicate fragments $N_S^{f,\text{rcv}}$:

$$R_{\text{msg,eff}}^{\text{cur,err}} = \frac{N_S^{\text{send,max}} - N_S^{f,\text{rcv}}}{N_S^{\text{send,max}}} \tag{12}$$

We can now extend eq. (11) by decreasing the number of remaining sending approaches applying this rate:

$$N_S^{f,\text{rem}} * R_{\text{msg,eff}}^{\text{cur,err}} \geq N_S^{f,\text{unacked}} \tag{13}$$

The relevance of this equation increases with being closer to the deadline, thus its evaluation should start after a minimum time duration $t^{\text{now}} - t_S^{\text{start}}$. If an upcoming violation is detected, the application level can be informed in order to adapt the driving behavior to a reduced set of up-to-date data, i.e. excluding the external data from sensor fusion and planing the driving behavior. As mentioned above, this fall-back strategy then complies to a fail-operational safety concept. In addition, we can also use this method on the sender side to stop transmission in advance in order to save channel resources.

D. Protocol parametrization

The three W2RP parameters t_f^{sh}, SRTT and S_f are determined in service discovery and stay fixed during operation. W.r.t. the example from section I, it's the choice of the parking lot owner to select the protocol parameters. The selection would be based on the number of concurrently parking cars, the properties of the wireless environment as well as the application requirements. Those characteristics correspond to our stack and application model parameters, and are therefore decisive for the allocation of resources to each individual vehicle w.r.t. W2RP parametrization. Adaption during protocol runtime would require cooperation between vehicles and the parking service adding uncertain timing to the existing protocol. Such uncertain additional timing would impact object transmission and possibly lead to spurious object deadline violations. This would hardly be acceptable under safety requirements.

V. Integrating W2RP into DDS/RTPS

The presented W2RP protocol is generally independent from a specific middleware. For example, W2RP can be used on a bare UDP/IP Stack without higher layer protocols. However, the integration into the popular DDS is helpful to investigate its behavior in a practically important application context.

Regarding the application interface, writing or reading samples to or from the database can be directly mapped to DDS's writer and reader interactions with their so-called **history caches**. Restricting their cache sizes to 1 reflects our $D_S = P_S$ requirement. As cache synchronization from a writer's to a reader's cache is based on the exchange of RTPS messages via UDP/IP we can adapt the RTPS structures and behavior for W2RP integration. The sizes of all structures referred to in the following are shown in Table I. Each RTPS

TABLE I: Protocol overhead of MAC, UDP, IP and RTPS

	MAC/ UDP/IP	**RTPS Header**	**RTPS NackFrag**	**RTPS HBFrag**	**RTPS DataFrag**
Size B (bytes)	36+20+8 = 64	20	28 + S_{bitmap}	28	36 + S_{payload}

message starts with an *RTPS Header*, followed by one or more submessages. Fragments are directly mapped to the *DataFrag* submessage. Moreover, the bitmaps are sent within so-called *NackFrag* submessages. For triggering *NackFrag* messages, an RTPS reader needs a fragment heartbeat submessage (*HBFrag*) each, which have to be sent together with every fragment. As HBFrag includes the highest yet sent FSN of a sample it directly transmits this information needed by W2RP.

As interoperability of RTPS is only based on the correct exchange of submessages, we have a high degree of freedom regarding timing and scheduling behavior, that can be exploited by a straightforward mechanism for shaping and timeout. As stated above, the NACK Guard is already included in RTPS, where it is called NACK suppression duration. Finally, the RTPS specification considers status storage objects as atomic units, thereby abstracting from the handling of individual data elements. Since the state representation has no impact on interoperability, we can stick to our FSO state model. Thus, based on the possibility of efficient DDS instrumentations based on W2RP we achieve high relevance.

VI. Evaluation

In this section, we evaluate the performance of the developed W2RP protocol w.r.t. the deadline violation rate and compare it with MAC layer and DDS reliability mechanisms. We start by showing a quantitative evaluation of sample communication with MAC layer protection, its efficiency, and

side effects. Thereafter, we combine the protected MAC layer with a state-of-the-art DDS middleware, as it would be done in a typical setup. The evaluation reveals performance degrading cross-effects between both reliability mechanisms. In detail, we show that in such a setup, a single highly disturbed DDS transmitter severely degrades the ability of other senders to hold their sample deadlines even if FERs have not changed. In contrast, we demonstrate that W2RP limits interference between any two vehicles. After a parameter discussion of W2RP, we compare it to standard DDS while disabling MAC layer retransmissions, which performs even worse in this case. Thus we can show, that all combinations of MAC layer and DDS retransmission mechanisms fail to address our use case, leaving W2RP as the only applicable protocol approach to a real-time reliable sample tranmission setup. Note, that we use standard DDS protocol overheads (cf. Table I) in all experiments.

The protocols discussed here, are implemented in Omnet++ and deployed on top of a UDP/IP- and WLAN-based communication stack. We use components from the widely used INET-package [37] for realistic UDP/IP and WLAN-based network simulation. The implementation of the physical channel as well as the MAC arbitration mechanism based on the models discussed in section III allow reproducible and comparable conditions for protocol evaluation. It is applied to all experiments in the same way to reach a fair evaluation. The models are sufficiently detailed to reflect the main WLAN protocol properties, and the physical channel is approximated by typical channel models found in literature [33], [34]. Future field tests are planned to include real world conditions. However, field tests are always influenced by standard interpretation of equipment providers and by the concrete physical environment.

A. Evaluation of MAC layer protection

As pointed out in section II, established MAC layers drop a frame after a maximum number of unsuccessful frame transmission attempts, irrespective of the data to be transmitted. This limitation of retries is essential and results from the MAC layer's role of providing application agnostic access to the radio medium, while limiting blocking effects on other data to be transmitted. Excessive retries would affect communication from the same sender to possibly other receivers (with a better connection) and would increase the load on a radio channel as a whole affecting other senders. We will later demonstrate that even a limited number of sample agnostic retries can seriously degrade a channel. Consequently, the frame drop rate is always greater than zero for a non-zero BER, which is shown in Fig. 4, for a maximum of 3 retransmissions and different FERs. Note, that frame fragmentation or aggregation (as in 802.11n [38]) do not protect against this frame dropping effect. Since a sample is dropped if any of its fragments is missing, the sample drop rate (cf. Fig. 4) is even higher than the frame drop rate. As a consequence, the MAC layer is incapable of sufficiently protecting sample transmissions against frame losses. Thus, we need a mechanism on a higher layer.

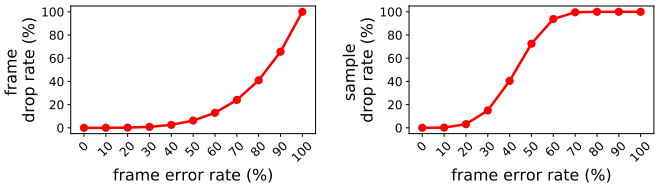


Fig. 4: Frame and sample drop rates for frame sizes of 1000 B and up to 3 retransmission attempts per frame. Samples are fragmented into 20 frames.

B. Pitfalls of standard DDS reliability mechanisms

As we pointed out in section II, TCP protects against data loss, but is not real-time capable even at moderate packet loss rates. Thus we evaluate standard DDS, which represents an automotive state-of-the-art middleware, but without adapting frame scheduling and retransmission to the sample deadline requirements as done by W2RP. In contrast to the proposed W2RP protocol, the current DDS specification uses a periodic heartbeat mechanism in order to trigger ACK-bitmaps at the reader. As there is no timeout available, a retransmission of a lost fragment relies on such a mechanism. Within a real-time setup, this approach **heavily relies on MAC layer retransmissions** to decrease the frame drop probability of this critical heartbeat and ACK messages. Moreover, after a sample has been pushed to the DDS history cache there is no time restriction to send fragments, thus, DDS creates burst transmissions of fragments. This is less critical at moderate FERs, however, at very high loss rates e.g. due to fading effects, MAC layer retransmissions of frames lead to a high number of frame transmission attempts, which can not be controlled or even noticed by the middleware layer. As a consequence high FERs combined with MAC layer retransmissions directly affect the transmission of other writers by injecting additional load and thus increasing the interference. In this paper, this effect is modelled by an increased $\bar{t}_a$ value.

We show this effect with the following example: One DDS writer W_{20} transmits samples with S_S of 20 kB and a fragmentation of 800 B. A second writer W_{10} transmits samples with S_S of 10 kB and a fragmentation of 1000 B. In both cases the FER is 50%. Furthermore, we assume a configuration of 802.11p so that W_{20} has an average arbitration time $\bar{t}_a$ of 800 µs and W_{10} of 1000 µs. The data rate is 27 Mbps. In this setup W_{10} and W_{20} are able to comply to their D_S requirements of 100 ms. If W_{10} is subject to significant fading effects and the FER increases, successful transmission is not possible anymore, however, the MAC layer floods the channel with frame retransmissions. Therefore, the maximum number of transmission attempts $N^{\max}_{\text{fr},10}$ in D_S is reached at a FER of appr. 86% (cf. eq. (5) and eq. (6)).

$$N^{\max}_{\text{fr},10} = \lfloor \frac{D_S}{\bar{t}_{b,10}} \rfloor = \lfloor \frac{D_S}{\bar{t}_{a,10} + d^{\text{msg}}_{\text{fr},10}} \rfloor = \lfloor \frac{100\,\text{ms}}{1000\,\mu\text{s} + 332\,\mu\text{s}} \rfloor = 75 \tag{14}$$

Note, that for 1 retransmission each DDS fragment, an average number of 3 MAC layer retransmissions already exeeds $N^{\max}_{\text{fr},10}$,

which is limited by the average blocking time $\bar{t}_b$. This is an increase of 55 transmissions in comparison to 20 at an FER of 50%. As a consequence, the additionally injected load by W_{10}, denoted as I_{10}^{add}, has to be added to $\bar{t}_a$ of W_{20}, whereby the arbitration protocol has to wait an additional Distributed Coordination Function Interframe Space (DIFS) of 20 µs before counting free slots at each interruption [18]:

$$\begin{aligned} I_{10}^{\text{add}} &= (N_{\text{fr},10}^{\max} - 20) * (\text{DIFS} + d_{f,10}^{\text{msg}}) \\ &= 55 * (20\,\mu\text{s} + 332\,\mu\text{s}) = 19\,360\,\mu\text{s} \end{aligned} \tag{15}$$

Effectively, the arbitration time of W_{20} over the duration of D_S is increased by I_{10}^{add} as it is the additional blocking time seen at the channel. Thus it reduces the maximum number of its arbitration attempts $N_{\text{f},10}^{\max}$ within a time span of D_S:

$$N_{\text{f},20}^{\max} = \lfloor \frac{D_S - I_{10}^{\text{add}}}{\bar{t}_{a,20} + d_{\text{fr},20}^{\text{msg}}} \rfloor = \lfloor \frac{100\,\text{ms} - 19\,360\,\mu\text{s}}{800\,\mu\text{s} + 272.6\,\mu\text{s}} \rfloor = 75 \tag{16}$$

From this we can determine the new average arbitration time of W_{20}, whereas we evenly divide the load interference I_{10}^{add} between $N_{\text{fr},20}^{\max}$ within D_S:

$$\bar{t}_{a,20}^{\text{new}} = \frac{I_{10}^{\text{add}}}{N_{\text{fr},20}^{\max}} + \bar{t}_{a,20} = \frac{19\,360\,\mu\text{s}}{75} + 800\,\mu\text{s} = 1058\,\mu\text{s} \tag{17}$$

Note, that this calculation is already sufficient to show the main effects of DDS's load interference. There are further negative effects from heartbeat transmissions that are omitted for simplicity. Fig. 5 shows two scenarios for W_{20}. The green curve represents the deadline violation rates of W_{20} in case that its $\bar{t}_a$ is unaffected from any other disturbing writer. The red curve shows the deadline violation rate in case, that W_{10} is affected by a high FER and requests retransmissions at maximum rate leading to an increased arbitration time, as approximated in eq. (17). In this case W_{20} violates D_S significantly at 50% FER, although he would have met the deadline without the increased interference at this value. Thus the

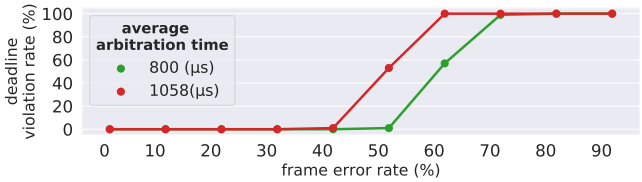


Fig. 5: Deadline violation rates of W_{20} at different FER and $\bar{t}_a$ values. At a FER of 50%, the degraded DDS writer (red curve) violates the sample deadline at high rate.

classic DDS approach is incapable of bounding interference between different writers, which is crucial for real-time critical applications sharing a channel. Hence, a single uncontrolled writer can make the whole channel unusable for other time-critical sample transmissions. Note that a closer investigation would need to consider cross dependencies between all arbitrations, but this would not give additional insights in the context of this paper.

In contrast to the uncontrolled load injection of the DDS heartbeat approach combined with MAC layer retransmissions, W2RP limits the load even at high FERs by application layer shaping and is able to manage retransmissions even without the use of MAC layer retransmissions. In an environment such as for automated valet parking with a known number of sample transmissions as well as fixed W2RP parametrization and application requirements, an upper average arbitration time based on the maximum *bounded* channel load can be calculated. Thus all sample transmissions are protected against excessive interference of a single and multiple writers. Consequently, we **disable MAC retransmissions** in the following and evaluate W2RP in comparison to DDS without such an uncontrollable low level reliability mechanism.

C. General parameterization

We start the evaluation of W2RP by defining a default parameterization shown in Table II. Considering the automated valet parking use case from section I, we set $P_S = D_S$ to 100 ms, which is a common sampling rate in such setups [9]. For S_S we assume preprocessed image data of size 20 kB. The channel access model is described by the average arbitration time w.r.t. eq. (1) and the IEEE 802.11p maximum data rate defined by the standard. We evaluate different FERs for which we can show relevant effects in their areas. Note, that denoted FERs are related to data frames and shorter ACK messages have a lower FER based on the bit error model (cf. eq. (2)). W2RP is configured by the fragment and bitmap size, the shaping time, the receiver delay time and the SRTT resulting from eq. (10) respectively. The protocol overheads include DDS submessage structures (cf. Table I). Message durations are related to the message sizes and the data rate. For all configurations the simulated timespan is 300 s, with 3000 sample transmissions. The signal path is 75 m.

TABLE II: Default values of all relevant parameters for the following experiments.

parameter name	symbol	value
sample period, sample deadline	P_S, D_S	P_S = D_S = 100 ms
sample size	S_S	20 kB
average arbitration time	$\bar{t}_a$	719.1 µs
frame error rate	$E_f^{E_b}$	0% to 90%
IEEE 802.11p data rate	R_b^p	27 Mbps
fragment size, bitmap size	S_f, S_{bitmap}	800 B, 4 B
shaping time	t_f^{sh}	varies
receiver delay time	t_d	500 µs
selected round trip time	$SRTT$	eq. (10)
protocol overheads (cf. Table I)	S_f^h, S_{ack}^h	148 B, 112 B
message sizes	S_f^{msg}, $S_{\text{ack}}^{\text{msg}}$	948 B, 116 B
message durations	d_f^{msg}, $d_{\text{ack}}^{\text{msg}}$	280.9 µs, 34.4 µs

D. Evaluation of basic W2RP parameters

The transmission behavior of W2RP can be adjusted by the fragment size S_f, the fragment shaping time t_f^{sh} and the selected round trip time SRTT. S_f is directly related to the FER based on the BER (cf. eq. (2)) so that smaller values are more favourable at higher BERs and only small data portions have to be retransmitted. However, smaller S_f values lead to

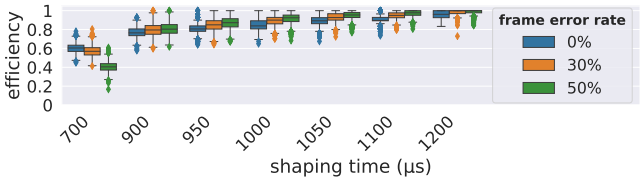


Fig. 6: Efficiency results for t_f^{sh} values around $\bar{t}_b = 1000\,\mu s$.

a higher amount of total fragments to be sent and thus also to more shaped arbitration attempts. Our latency results for a $t_f^{sh} = 1200\,\mu s$ for varying fragment sizes indicate a latency optimum for our configuration at about $S_f = 800\,B$ at high error rates. This underlines the mentioned trade-offs.

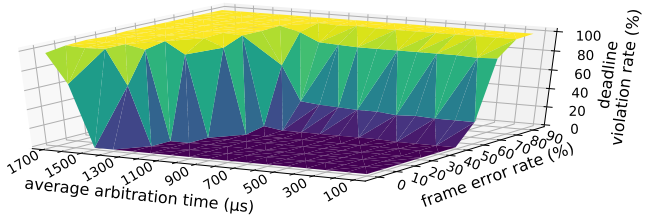
As we identified shaping of $\bar{t}_b$, and thus of the RTT, as critical for efficient fragment loss identification, we analyse the efficiency over varying t_f^{sh} and different FERs. We also couple the SRTT to these variations (cf. eq. (10)). Fig. 6 shows, that for t_f^{sh} values above $\bar{t}_b$ (1000 µs) the efficiency is rapidly approaching 1. Statistical variations and ACK losses related to timeout retransmissions produce only single unnecessary retransmissions at the end of sample transmission, moreover, samples might even already been fully transmitted. t_f^{sh} values below $\bar{t}_b$ rapidly decrease the efficiency, as a consequence of MAC buffer queuing effects. Our latency evaluation shows, that inefficiency at low t_f^{sh}'s increases sample latency more than larger shaping times at higher t_f^{sh}'s. Results underline our assumption, that t_f^{sh} represents a limit to the maximum manageable channel load modeled by $\bar{t}_a$, since $t_f^{sh} - d_f^{\text{msg}} \leq \bar{t}_a$ reduces the efficiency significantly. Increasing t_f^{sh} leads to high efficiency even at high blocking times, but limits the number of fragment retransmissions. Decreasing t_f^{sh} allows to handle higher FERs under the condition of lower blocking times, i.e. a lower channel load. Both bounds can be estimated based on the technical limits described in section IV-A. Evaluations on the SRTT configuration has shown, that SRTT configuration based on eq. (10) keeps timeout delays at the end of sample transmission sufficiently small.

As a main motivation of W2RP is to exploit sample deadlines for frame-level scheduling, further evaluation has shown that efficiency increases with larger S_S values, which underlines our protocol approach. Consequently, larger samples stick more tightly to the technical bounds.

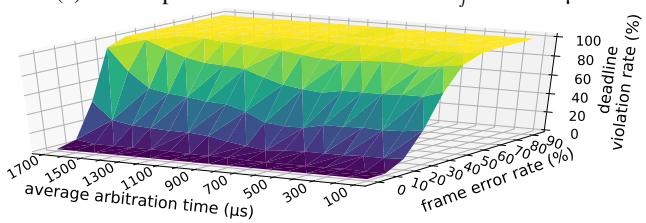
E. Configuration of W2RP and comparison with DDS

When adapting W2RP protocol parameters, we can show that high S_f values are more favourable at lower BERs as less arbitration attempts are needed (w.r.t. eq. (2)). However, at higher BERs, there is an S_f value optimizing the sample latency, as lower S_f values lead to lower FERs and higher S_f values reduce the number of arbitration attempts. Hence, t_f^{sh} is the central parameter to configure W2RP towards a feasible channel model, which is also the base for optimal S_f estimation. On the one hand, it bounds the maximum tolerable arbitration time in the context of the effective blocking time $\bar{t}_{b,\text{eff}}$, as the protocol loses efficiency for $t_f^{sh} \leq \bar{t}_b$. On the other hand, smaller t_f^{sh} values are needed in order to enable a high retransmission rate at high FERs. As the third parameter, SRTT can be configured based on eq. (10).

The parameters $\bar{t}_a$ and BER, which we translate to the FER, model the capabilities of a channel and hence yield the technical limits $R_{\text{msg}}^{\text{max,err}}$ and D_S^{min} for a given sample transmission. The limiting factor to approach the technical limits, however, is the deadline of a sample. Fig. 7a shows the evaluation of the D_S violation rate for different values of $\bar{t}_a$ and FER and a t_f^{sh} configuration of 1400 µs. This configuration can provide D_S up to a FER of 50% with only few violations at 60% FER for arbitration times, which are below W2RP's critical threshold $\bar{t}_a = t_f^{sh} - d_{\text{fr}}^{msg} = 1119.1\,\mu s$, from where W2RP loses efficiency. Note, that the technical bound on the maximum handable loss rate $R_{\text{msg}}^{\text{max,err}}$ is 66% for this example configuration. Consequently, t_f^{sh} can be decreased in order to guarantee D_S at higher FERs with the trade-off of a lower acceptable average arbitration time. This allows W2RP to be adapted to specific load and error scenarios, such as a parking lot in our automated valet parking use case. Due to the shaping, the number of fragment (re)transmissions is limited, which bounds the interference between multiple sample transmissions and enables to pre-estimate certain setups.



(a) W2RP performance evaluation with $t_f^{sh} = 1400\,\mu s$.



(b) Without MAC layer retransmissions, standard DDS can meet the deadline only under very low FERs.

Fig. 7: Comparison of W2RP and standard DDS, both with MAC layer retransmissions disabled.

In contrast, the standard DDS transmission without support of MAC layer protection cannot guarantee deadlines even at low error rates, as shown in Fig. 7b. The reason for this is that without MAC layer retransmissions periodic heartbeats or bitmaps are likely to be dropped. Thus the protocol stalls until the next heartbeat, which is unfavorable in a real-time environment. Hence, decreasing the heartbeat period is even worse, as this leads to more arbitration attempts without actually transmitting data. This is an interesting and important result for DDS QoS, as it shows that DDS protection is ineffective for lossy communication in real-time applications.

If we turn on MAC layer retransmissions, transmission performs slightly worse than W2RP, as long as there is no interferer with high MAC level retransmission load ($\bar{t}_a = 800\,\mu s$). This changes if only a single interferer increases the arbitration time to $\bar{t}_a = 1058\,\mu s$. This confirms what we already saw in section VI-B: the combination of DDS with application agnostic MAC layer retransmissions is not robust in real-time applications and quickly loses efficiency at higher frame error rates. In contrast, the W2RP protocol controls error correction based on application knowledge, thereby avoiding excessive retransmissions under higher error rates that only cause channel load, but do no improve reliability.

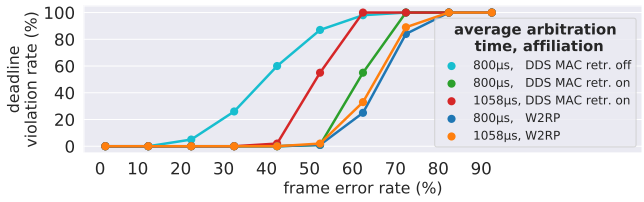


Fig. 8: Comparison of all evaluated scenarios.

There is a caveat: In our evaluation, we assumed that all traffic uses the same protocol. If other senders spoil the channel with excessive privileged retransmissions, then a sender using the W2RP protocol will also lose efficiency. So, a valet parking service should require access shaping, as in W2RP, to improve the service.

F. Comments on W2RP overhead

Since data structure and messaging are the same as in DDS, there is no additional memory overhead compared to standard DDS, both in sender and receiver. The computational overhead is limited to straigtforward packet reordering. The practically most relevant W2RP wireless protocol overhead consists of few more acknowledgment messages. In all experiments in this paper, the total channel overhead is between 1% and 6%, which is acceptable with respect to the gained real-time and reliability properties.

VII. Conclusion

Current V2X communication is optimized for cooperation scenarios that require exchange of small data sets in one or few packets. Real-time communication of larger data objects, such as for collaborative sensing, is on the roadmaps, but needs further research to address the conflicting requirements of real-time transmission of large data objects over a lossy multi-access channel. This paper presented W2RP, a UDP-based protocol that exploits the larger application-level deadline of samples for efficient error correction under integrity guarantees. It is targeted to be used in combination with popular application and communication standards. For this paper, we selected DDS as middleware for data centric embedded systems, and IEEE 802.11p as an established V2X standard. A typical use case, automated valet parking augmented by infrastructure cameras, was taken to explain the challenges and investigate the proposed solution. The vehicle has two modes, an augmented mode using the infrastructure cameras in sensor fusion enabling increased vehicle speed, and a slower safety mode using in-vehicle sensors only. The target was to keep the augmented mode as long as possible, but safely and immediately detect a failing or outdated sensor communication. The proposed W2RP protocol was evaluated against existing solutions, both on the DDS application layer and on the MAC layer as well as combinations thereof. We showed that MAC layer retransmissions of DDS data under higher frame loss rates leads to excessive channel load interfering with other vehicles thereby likely forcing them into safety mode. Using the DDS error protection with no MAC layer transmission can avoid interference, but leads to data age violation even under moderate frame loss rates. Then, we demonstrated that W2RP outperforms the existing solutions by exploiting the combined knowledge of application requirements and communication channel properties. The results are not limited to the use case but can be applied to similar cases of DDS and wireless communication. The results can in particular be applied to newer standards, such as 802.11n, that offer higher data rates and frame aggregation for shorter average access times, but use MAC layer protocols with the same limitations.

Acknowledgment

This work is partly funded by the project ”UNICARagil” (FKZ EM2ADIS002). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).

References

[1] T. Li, Q. Ni, D. Malone, D. Leith, Y. Xiao, and T. Turletti, “Aggregation with fragment retransmission for very high-speed WLANs,” *IEEE/ACM transactions on networking*, vol. 17, no. 2, pp. 591–604, 2009.

[2] X. Zhou and A. Boukerche, “Aflas: An adaptive frame length aggregation scheme for vehicular networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 855–867, 2016.

[3] X. Jin, C. Xia, H. Xu, J. Wang, and P. Zeng, “Mixed criticality scheduling for industrial wireless sensor networks,” *Sensors*, vol. 16, no. 9, p. 1376, 2016.

[4] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, “Real-time scheduling for wirelesshart networks,” in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 150–159.

[5] A. Burns, J. Harbin, L. Indrusiak, I. Bate, R. Davis, and D. Griffin, “Airtight: A resilient wireless communication protocol for mixed-criticality systems,” in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2018, pp. 65–75.

[6] Object Management Group (OMG), “Data Distribution Service, Version 1.4,” OMG Document Number formal/2015-04-10, March 2015.

[7] “Bosch: Automated Valet Parking,” https://www.bosch.com/stories/automated-valet-parking, accessed: 2020-09-17.

[8] V. Schönemann, M. Duschek, and H. Winner, “Maneuver-based adaptive safety zone for infrastructure-supported automated valet parking,” 01 2019, pp. 343–351.

[9] M. Kneissl, A. K. Madhusudhanan, A. Molin, H. Esen, and S. Hirche, “A multi-vehicle control framework with application to automated valet parking,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[10] ISO, “26262: Road vehicles-functional safety,” 2018.

[11] SAE, “J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” 2018.

[12] Y. Wang, G. de Veciana, T. Shimizu, and H. Lu, “Performance and scaling of collaborative sensing and networking for automated driving applications,” in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[13] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *IEEE Communications Surveys & Tutorialsi*, 2021, preprint.

[14] SAE, "J3216: Taxonomy and Definitions for Terms Related to Cooperative Driving Automation for On-Road Motor Vehicles," 2020.

[15] *AUTOSAR - Specification of Communication Management*, Adaptive Platform R19-11 ed., AUTOSAR GbR, November 2019.

[16] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[17] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[18] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, 2010.

[19] *ETSI TS 102 637-2: "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service"*, ETSI, 2011, v1.2.1 (2011-03).

[20] SAE, "J2735: Dedicated Short Range Communications (DSRC) Message Set Dictionary," 2016.

[21] Cheng Peng Fu and S. C. Liew, "Tcp veno: Tcp enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[22] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "Tcp westwood: Bandwidth estimation for enhanced transport over wireless links," vol. 8, 01 2001, pp. 287–297.

[23] G. Lv, X. Xu, K. Su, and Q. Chen, "Uap: A new udp-based application level transport protocol," in *2011 Second International Conference on Networking and Distributed Computing*, 2011, pp. 3–6.

[24] X. Shi, G. Tian, and Y. Tian, "A reliable real-time transport protocol for control systems over wireless networks," in *Australasian Telecommunication Networks and Applications Conference (ATNAC) 2012*, 2012.

[25] "Ofera Project: Open Framework for Embedded Robot Applications," http://www.ofera.eu/, accessed: 2020-03-30.

[26] H.-Y. Choi, A. L. King, and I. Lee, "Making DDS really real-time with OpenFlow," in *2016 International Conference on Embedded Software (EMSOFT)*, Oct. 2016.

[27] A. A. Al-Roubaiey, T. R. Sheltami, A. S. H. Mahmoud, and K. Salah, "Reliable middleware for wireless sensor-actuator networks," *IEEE Access*, vol. 7, pp. 14 099–14 111, 2019.

[28] J. Schmitt, S. Bondorf, and W. Y. Poe, "The sensor network calculus as key to the design of wireless sensor networks with predictable performance," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, p. 21, 2017.

[29] S. Das, P. Kar, and S. Barman, "Analysis of ieee 802.11 wlan frame aggregation under different network conditions," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2017, pp. 1240–1245.

[30] B. Gautam, "Amsdu vs ampdu: A brief tutorial on wifi aggregation support," Tech. Rep, Tech. Rep., 2017.

[31] D. Wang, R. R. Sattiraju, A. Qiu, S. Partani, and H. D. Schotten, "Effect of retransmissions on the performance of c-v2x communication for 5g," *arXiv preprint arXiv:2007.08822*, 2020.

[32] F. Klingler, F. Dressler, and C. Sommer, "Ieee 802.11p unicast considered harmful," in *2015 IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 76–83.

[33] Y. Zang, L. Stibor, G. Orfanos, S. Guo, and H.-J. Reumerman, "An error model for inter-vehicle communications in highway scenarios at 5.9 ghz," in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, 2005, pp. 49–56.

[34] C. A. Gómez-Vega, J. J. Jaime-Rodríguez, C. A. Gutiérrez, and R. Velázquez, "Bit error rate performance analysis of vehicular communication systems considering velocity variations of the mobile stations," in *2017 IEEE 37th Central America and Panama Convention (CONCAPAN XXXVII)*, 2017, pp. 1–6.

[35] S. Eichler, "Performance evaluation of the ieee 802.11 p wave communication standard," in *2007 IEEE 66th Vehicular Technology Conference*. IEEE, 2007, pp. 2199–2203.

[36] Object Management Group (OMG), "DDS Interoperability Wire Protocol, Version 2.3," OMG Document Number formal/2019-04-03, May 2019.

[37] "INET - Framework," https://inet.omnetpp.org/, accessed: 2020-03-21.

[38] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. 1–565, 2009.